Teoria della NP-completezza

Gianluigi Zavattaro
Dip. di Informatica – Scienza e Ingegneria
Università di Bologna
gianluigi.zavattaro@unibo.it

Slide realizzate a partire da materiale fornito dal Prof. Moreno Marzolla

Problemi computazionali

- Un problema Q può essere definito come una relazione $Q \subseteq I \times S$
  - I è l'insieme delle istanze di ingresso
  - S è l'insieme delle soluzioni
- Possiamo immaginare Q come un predicato che, dato in ingresso una istanza di input $x \in I$ e una soluzione $s \in S$, restituisce:
  - 1 se $(x,s) \in Q$ (s è soluzione del problema Q sull'istanza x)
  - 0 altrimenti (s non è soluzione del problema Q sull'istanza x)

Decidere, ricercare, ottimizzare

- Problemi di decisione
  - Problemi che richiedono una risposta binaria (S = {0,1})
  - Es: “Un dato grafo *x* è connesso?”
  - Es: “Un elemento *x* è contenuto in un dizionario?”
- Problemi di ricerca
  - Data una istanza x, restituire una soluzione *s* tale che $(x,s) \in Q$
  - Es: “Trovare un albero di copertura per il grafo *x*”
- Problemi di ottimizzazione
  - Data una istanza *x*, restituire la “migliore” soluzione *s*
  - Es: “Trovare un minimo albero di copertura per il grafo *x*”

Classi di complessità

- Data una funzione $f(n)$, chiamiamo TIME($f(n)$) (risp. SPACE($f(n)$)) l'insieme di tutti i problemi decisionali che possono essere risolti in tempo (risp. in spazio) O($f(n)$)
  - Ossia tutti i problemi che ammettono un algoritmo A t.c.:
    - per ogni istanza di input x, A restituisce 1 se e solo se $(x,1) \in Q$
    - A ha costo O($f(n)$) in tempo (risp. in spazio)

Classi di complessità

- La classe P è la classe dei problemi risolvibili in tempo polinomiale nella dimensione *n* dell'istanza di ingresso

$$P = \cup_{c=0}^{\infty} TIME\left(n^c\right)$$

- La classe PSPACE è la classe dei problemi risolvibili in spazio polinomiale nella dimensione *n* dell'istanza di ingresso

$$PSPACE = \cup_{c=0}^{\infty} SPACE\left(n^c\right)$$

Classi di complessità

- La classe EXPTIME è la classe dei problemi risolvibili in tempo esponenziale nella dimensione *n* dell'istanza di ingresso

$$EXPTIME = \cup_{c=0}^{\infty} TIME\left(2^{n^c}\right)$$

Classi di complessità

- Un algoritmo che richiede tempo polinomiale riuscirà al più ad accedere ad un numero polinomiale di locazioni di memoria diverse, quindi:

$$P \subseteq PSPACE$$

- Poiché n^c locazioni di memoria possono trovarsi al più in 2^{ n^c } stati diversi, si ha anche

$$PSPACE \subseteq EXPTIME$$

- Non è noto se le inclusioni di cui sopra sono strette (non si sa se P ⊂ PSPACE o se PSPACE ⊂ EXPTIME), ma una delle due inclusioni è stretta! (in quanto si sa che P ⊂ EXPTIME)

Esempi

- Ogni espressione booleana si può trasformare in forma normale congiuntiva (congiunzione di clausole, dove una clausola è una disgiunzione di letterali, e un letterale è una variabile o una variabile negata)
  - Esempio: $(x \vee \bar{y} \vee z) \wedge (y \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$
- *Data una espressione booleana in forma normale congiuntiva, il problema della soddisfacibilità (SAT) richiede di verificare se esiste una assegnazione di valori alle variabili che rende l'espressione vera*
- Il problema della soddisfacibilità è in PSPACE (e quindi anche in EXPTIME)
  - Se ci sono *n* variabili, basta tentare tutte le 2^n assegnazioni
  - Questo si fa con un algoritmo che richiede spazio lineare

Verificare vs Certificare

- Come visto in precedenza, nei problemi di decisione siamo interessati a sapere se una istanza *x* del problema verifica una certa proprietà
  - Es: L'espressione booleana in forma normale congiuntiva K è soddisfacibile?
- Spesso però siamo anche interessati a conoscere un qualche oggetto *y*, che dipende da *x* e dal problema da risolvere, che possa certificare il fatto che *x* gode di tale proprietà
  - Es: una assegnazione di variabili che rende l'espressione booleana K vera

Esempio

- Il problema delle formule booleane quantificate (tutte le variabili sono quantificate esistenzialmente o universalmente) richiede di verificare se una certa formula booleana quantificata è vera
- Es: $\exists x \forall y \exists z \forall w : (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee w) \wedge y$
  - l'espressione sopra è falsa. Perché?
- Per il problema delle formule booleane quantificate **non** conosciamo certificati di dimensione polinomiale!
  - Quando l'espressione è vera, se le variabili quantificate universalmente sono *k* possiamo fornire un numero *esponenziale* (2^k) di possibili assegnamenti per certificare che l'espressione è vera!

Definizione

- Informalmente NP è la classe dei problemi decisionali che ammettono certificati verificabili in tempo polinomiale
  - Più precisamente, dato un problema $Q \in NP$ esiste un algoritmo decisionale polinomiale D tale che:
    - Per ogni istanza x
      esiste un certificato di dimensione polinomiale C_x tale che $D(C_x)=1$ se e solo se $(x,1) \in Q$
  - Algoritmo "forza bruta" per problemi NP
    - Genero tutti i possibili certificati polinomiali C, e controllo se D(C)=1
  - Sarebbe bello avere aiuto da un "**indovino**" che ci indica, se esiste, quale è il certificato giusto da verificare!

Non determinismo

- Negli algoritmi visti a lezione, ogni passo è sempre univocamente determinato dallo stato delle variabili
- Un algoritmo decisionale **non deterministico**, invece, oltre alle normali istruzioni può eseguire istruzioni del tipo "indovina $z \in S$"
  - In altre parole, può indovinare un valore "corretto" e far proseguire la computazione nella "giusta" direzione
  - Il non determinismo è il nostro "indovino"!

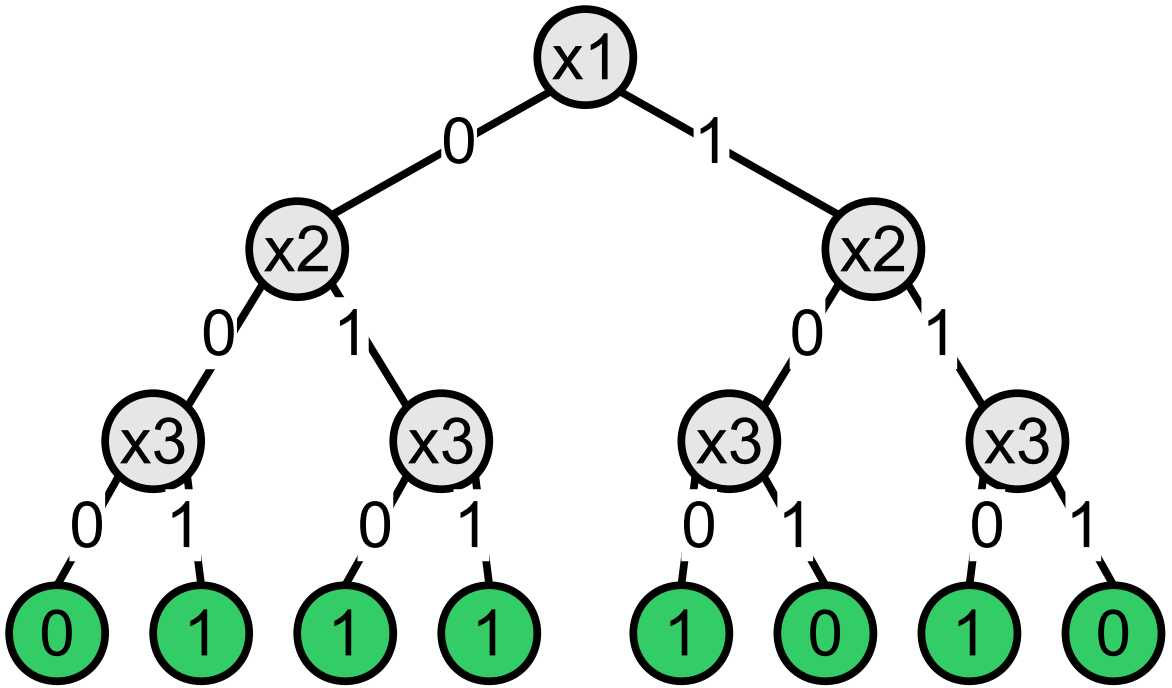
Soddisfacibilità non deterministica

- Usando il **non determinismo** si può risolvere il problema della soddisfacibilità in tempo lineare:
  - Si esegue una sequenza di istruzioni indovina $x_i \in \{0,1\}$ per ogni variabile x_i e poi si controlla che l'espressione sia vera
- **Nota**: Un algoritmo non deterministico può essere rappresentato da un albero di decisione
  - l'algoritmo restituisce 1 se c'è almeno una foglia che restituisce 1, altrimenti restituisce 0

Esempio

- L'espressione seguente è soddisfacibile?

$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3)$$

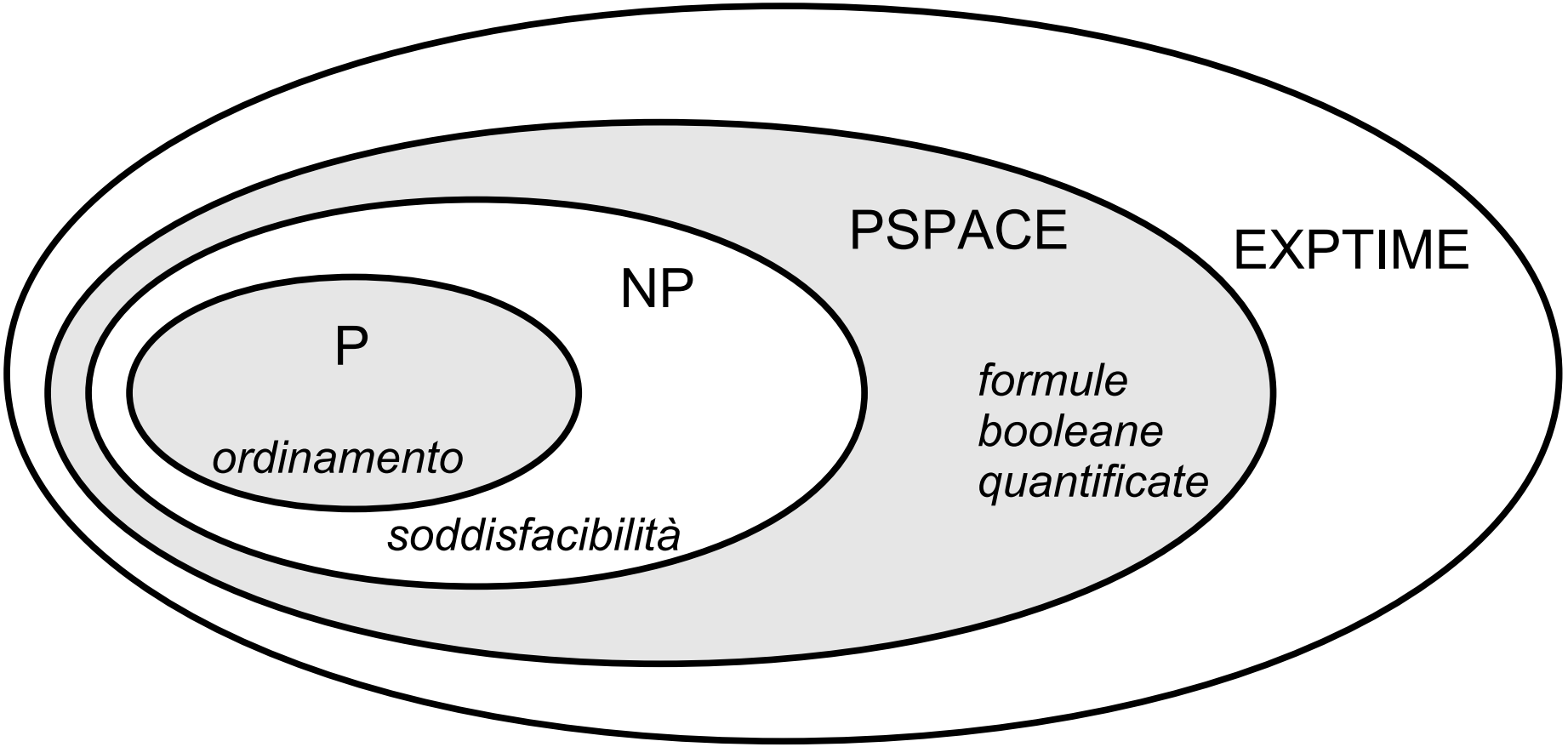


Definizione

- Data una funzione f(n), chiamiamo NTIME(f(n)) l'insieme dei problemi decisionali che possono essere risolti da un algoritmo non deterministico in tempo O(f(n)). La classe NP è la classe dei problemi risolvibili in tempo polinomiale non deterministico nella dimensione n dell'istanza di ingresso:

$$NP = \cup_{c=0}^{\infty} NTIME\left(n^c\right)$$

Uno sguardo alla gerarchia



- Delle inclusioni qui sotto almeno una è propria:

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

- Si **congettura** che le inclusioni siano tutte proprie

Riducibilità polinomiale

- Consideriamo due problemi decisionali
  - $Q1 \subseteq I1 \times \{0,1\}$ e
  - $Q2 \subseteq I2 \times \{0,1\}$
- Supponiamo di avere una funzione $f: I1 \rightarrow I2$ in grado di trasformare—in tempo polinomiale—istanze di input per Q1 in istanze di input per Q2, tali che per ogni soluzione s

  $(x,s) \in Q1$ se e solo se $(f(x), s) \in Q2$
- Allora diremo che:

  Q1 è riducibile polinomialmente a Q2

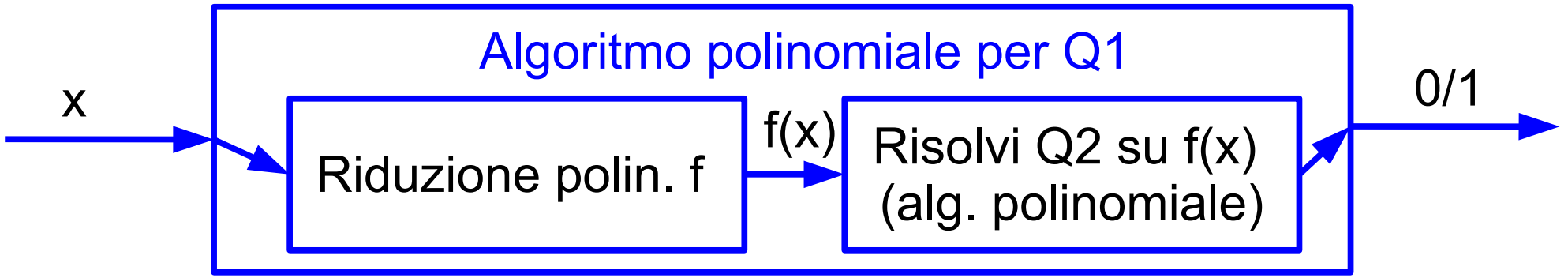
Esempio di riducibilità polinomiale

- La soddisfacilità di espressioni booleane è riducibile polinomialmente nella verifica di verità di formule booleane quantificate
  - Consideriamo l'espressione booleana E che contiene le variabili x_1, .. x_n
  - Consideriamo f tale che restituisce una formula booleana quantificata così definita: $f(E) = \exists x_1 . \ .. \ . \exists x_n . E$
  - Tale trasformazione ha costo lineare e abbiamo che:

    E è **soddisfacibile** se e solo se $f(E) = \exists x_1 . \ .. \ . \exists x_n . E$ è **vera**
- **Non si sa** se esiste una riduzione polinomiale in senso inverso (da verità formule booleane quantificate alla soddisfacibilità di espressioni booleane)

Implicazioni della riducibilità polinomiale

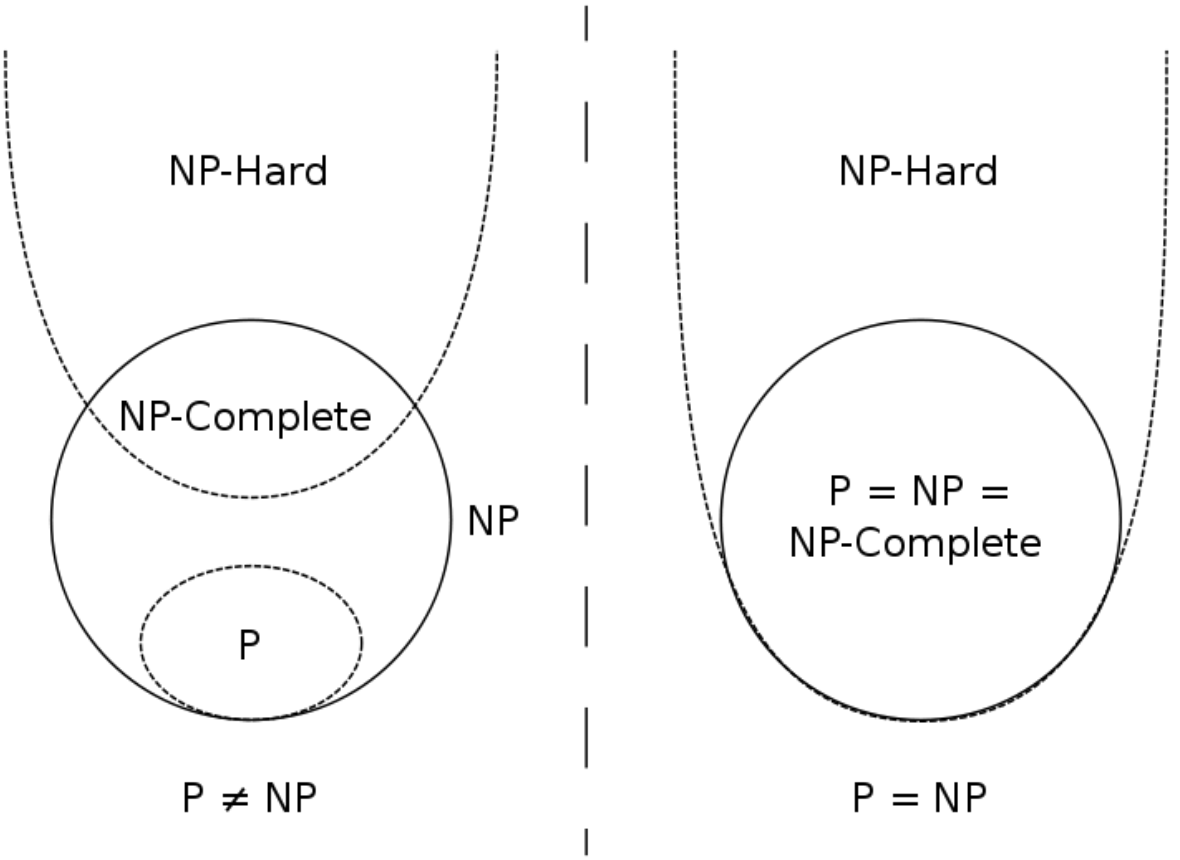
- Consideriamo un problema Q2 per il quale sia noto un algoritmo risolutivo polinomiale
  - Quindi $Q2 \in P$
- Supponiamo che Q1 sia riducibile polinomialmente a Q2
  - Allora anche $Q1 \in P$
  - Per risolvere Q1 su istanza x, basta trasformare (in tempo polinomiale x nella relativa istanza f(x) per Q2), e poi usare l'algoritmo risolutivo (polinomiale) che risolve Q2



NP-completezza

- Un problema decisionale Q si dice NP-arduo se ogni problema W $\in$ NP è riducibile polinomialmente a Q
- Un problema decisionale Q si dice NP-completo se appartiene alla classe NP ed è NP-arduo
- **Nota**: se un qualunque problema decisionale NP-completo appartenesse alla classe P, allora P = NP
  - Sarebbe un disastro!
    - Il problema della decifratura di un documento crittografato sarebbe polinomiale (quindi eseguibile in tempi “ragionevoli”)
    - Infatti, se l'algoritmo di cifratura (polinomiale con chiave di cifratura) è noto, allora esiste un certificato polinomiale per il problema della decifratura: la password di cifratura!

Graficamente



Esempio di problema NP-completo

- Problema della fermata limitata:
  - Dati un programma X ed in intero k, restituisce 1 se esiste un input per X che esegue al più k operazioni (0 altrimenti)
- La fermata limitata è in NP
  - Dato un programma X ed un intero k, il certificato è l'eventuale input y che termina in al più k passi (basta eseguire al più k passi di X sull'input y)
- La fermata limitata è un problema NP-arduo
  - Consideriamo un problema $Q \in NP$ ed una sua istanza x
    - Sia C_x l'eventuale certificato controllabile in p(|x|) passi (p polinomio)
  - Considero il seguente programma X
    - Considero l'input come un certificato per istanza x di Q, se il certificato fallisce va in loop infinito, altrimenti termina
  - Si ha che $(x,1) \in Q$ sse X ha fermata limitata in p(|x|) passi

Teorema di Cook

- **Teorema di Cook**: Il problema SAT (soddisfacibilità di espressioni booleane in forma normale congiuntiva) è NP-completo
  - La dimostrazione si basa su una riduzione del problema della fermata limitata in una espressione booleana in forma normale congiuntiva di dimensione polinomiale
- **Corollario**: Dato un problema Q, se esiste una riduzione di SAT in Q, allora possiamo concludere che Q è un problema **NP-arduo**

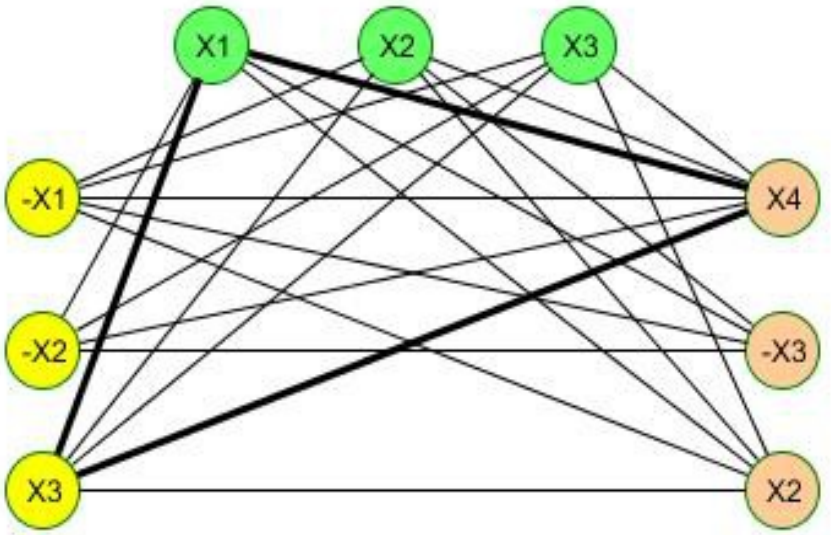
Un altro problema NP-completo

- **Problema della clique di dimensione k**:
  - Dato un grafo G verificare se esiste un sottografo completo di k vertici (un arco per ogni coppia di vertici)
- Il problema è in **NP**: il certificato è l'insieme di k vertici
- Il problema è **NP-arduo**: si riduce SAT nel problema della clique
  - ogni clausola genera un vertice per ogni letterale
  - due vertici di due diverse clausole sono collegati da un arco se **non** sono letterali incompatibili
    - **non** sono la stessa variabile, una negata, una no
  - l'espressione è soddisfacibile **sse** il grafo contiene una clique di dimensione pari al numero di clausole

Un altro problema NP-completo

- Un esempio di riduzione da SAT a problema della clique

$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_4 \vee \bar{x}_3 \vee x_2)$$



  - La clique indica anche validi assegnamenti:

$$x_1 = true\ ,\ x_2 = true / false\ ,\ x_3 = true\ ,\ x_4 = true$$

Problemi NP-completi

- Esistono tanti problemi interessanti NP-completi
  - **Commesso viaggiatore**:
    dato un grafo pesato completo ed un valore *k*, decidere se esiste un cammino semplice che copre tutti i vertici di peso inferiore a *k*
  - **Bin packing problem**:
    dati *n* oggetti, ognuno di peso *p[i]*, e *k* contenitori di capacità *c*, decidere se c'è modo di distribuire tutti gli *n* oggetti nei *k* contenitori senza eccedere la capacità *c*
  - **Sudoku**:

    Data una matrice di dimensione n^2 x n^2 , organizzata in blocchi di celle di dimensione n x n, con alcune celle contenenti numeri compresi fra 1 e n^2, decidere se è possibile completare la matrice in modo tale che ogni riga, colonna e blocco contenga tutti i numeri compresi fra 1 e n^2

Conclusioni

- I problemi NP-completi rappresentano il confine fra i problemi trattabili (risolvibili in tempi “ragionevoli” anche su input di dimensioni non banali) ed i problemi non trattabili
- Molti problemi di interesse pratico sono NP-completi
- Per questo motivo c'è forte interesse nello studio di questa classe di problemi
  - Sentirete parlare di problemi NP-completi anche in insegnamenti dei prossimi anni (es. Ottimizzazione Combinatoria, Informatica Teorica, Sicurezza, Crittografia, ...)